CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Elsevier - Publisher Connector

Available online at www.sciencedirect.com
ScienceDirect
International Journal of Approximate Reasoning
45 (2007) 470–487

INTERNATIONAL JOURNAL OF APPROXIMATE REASONING
www.elsevier.com/locate/ijar

A study on the modeling ability of the IDS method: A soft computing technique using pattern-based information processing

Masayuki Murakami *, Nakaji Honda

Department of Systems Engineering, The University of Electro-Communications, Chofu, Tokyo 182-8585, Japan

Received 3 November 2005; received in revised form 16 March 2006; accepted 30 June 2006
Available online 10 October 2006

Abstract

The ink drop spread (IDS) method is a modeling technique developed by algorithmically mimicking the information-handling processes of the human brain. This method has been proposed as a new approach to soft computing. IDS modeling is characterized by processing that uses intuitive pattern information instead of complex formulas, and it is capable of stable and fast convergences. This paper investigates the modeling ability of the IDS method based on three typical benchmarks. Experimental results demonstrated that the IDS method can handle various modeling targets, ranging from logic operations to complex nonlinear systems, and that its modeling performance is satisfactory in comparison with that of feedforward neural networks.
© 2006 Elsevier Inc. All rights reserved.

Keywords: IDS method; Ink drop spread; Modeling ability; Soft computing

1. Introduction

As engineering deals with complex real-world problems, computational techniques termed "soft computing" gains importance. Soft computing is a discipline that involves

* Corresponding author. Tel.: +81 42 443 5800; fax: +81 42 443 8020.
E-mail addresses: murakami@ieee.org (M. Murakami), honda@se.uec.ac.jp (N. Honda).
URL: http://www.fs.se.uec.ac.jp/~masayuki (M. Murakami).

0888-613X/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.ijar.2006.06.022

low-cost, tractable, and robust computing in the presence of uncertainty [1]. Humans possess a remarkable ability to process intricate information with ease; it is difficult to model such information by classical mathematical approaches. Therefore, some soft computing methodologies focus on human thought processes and physical characteristics. For example, fuzzy logic is modeled on the linguistic and logical aspects of the human thought processes, and neural networks are physically modeled on the human brain. From the perspective of both software and hardware, neural networks might possess maximum potential for use in various industrial applications. This may be due to the fact that neural networks are modeled on the human brain that has flexible information-processing abilities, and are capable of high-speed computation due to heavy parallel processing on a dedicated platform.

Although neural networks exhibit many attractive features, they involve several drawbacks. With regard to the modeling technique to be discussed in this paper, we refer to two problems that typically occur in neural network applications. The first problem is that a considerable amount of time is required to train neural networks. This becomes a disadvantage in applying them to real-time and dynamic learning applications. The second problem is that they are considered similar to a black box. It is difficult to obtain knowledge embedded in trained neural networks and provide it to a user in a comprehensible form [2]. When any model is applied to industrial systems, users should understand specific rules that describe its internal behavior. In particular, users of safety-critical systems may hesitate to use a black-box-like tool such as neural networks.

We have attempted to establish the active learning method (ALM) [3–5] as a soft computing methodology. It is analogous to fuzzy logic since it is algorithmically modeled on intelligent information-handling processes of the human brain. However, its processing nature is similar to that of neural networks rather than fuzzy logic. While fuzzy logic uses linguistic and logical processing, the ALM is characterized by intuitive pattern-based processing. This concept is based on the hypothesis that humans interpret information in the form of pattern-like images rather than as numerical or logical forms.

The ALM uses its modeling technique termed the ink drop spread (IDS) method. This paper compares the IDS method with feedforward neural networks (FNNs) in terms of their algorithmic characteristics and modeling abilities. The IDS method has advantages over standard neural networks with respect to the above-mentioned drawbacks. First, it does not require intricate calculations, randomly initialized parameters, and the iteration of the same training set observed in neural network learning. Thus, it is possible to obtain stable fast convergences [6,7]. Second, the features of a modeling target are stored in the form of easily comprehensible pattern information and the model output is computed using learning data obtained from this information, based on a simple combination rule set. This will help a user to understand how an IDS model analyzes the target system and utilizes learning data for prediction. In addition to these advantages, a good modeling ability of the IDS method will make it a very useful soft computing tool. Therefore, we investigated the modeling ability of the IDS method. In this study, we used three typical benchmarks: the XOR problem, regression problem of Hwang's five-function set, and two-spiral problem. Experimental results demonstrated that the IDS method can deal with various modeling targets, ranging from logic operations to complex nonlinear systems, and its modeling performance is satisfactory in comparison with that of several FNNs.

2. IDS method

2.1. Outline of the algorithm

In IDS modeling, a multi-input single-output (MISO) system is split into single-input single-output (SISO) systems, as shown in Fig. 1. This implies that a complex system is broken down into simpler aspects to acquire information in a more comprehensible form. Further, this is similar to the manner in which humans interpret complicated subjects. Each SISO is referred to as an IDS unit; it functions as a modeling engine to elicit a feature from the system to be modeled.

In order to explain the modeling process of the IDS method, we use a two-input system that has an input/output relationship, as shown in Fig. 2. Based on the concept shown in Fig. 1, we convert the entire input/output system into four SISOs by dividing each input domain into two partitions, as shown in Fig. 3. In order to develop a certain feature within

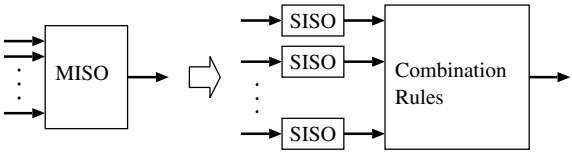


Fig. 1. Modeling approach adopted in the IDS method.

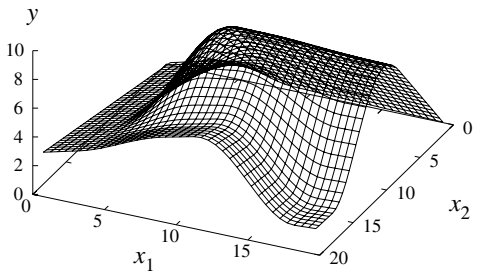


Fig. 2. Input/output relationship of a system.

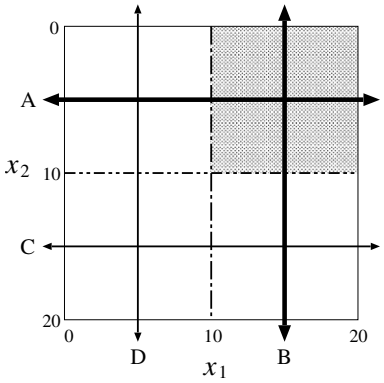


Fig. 3. Partitioning of input domains.

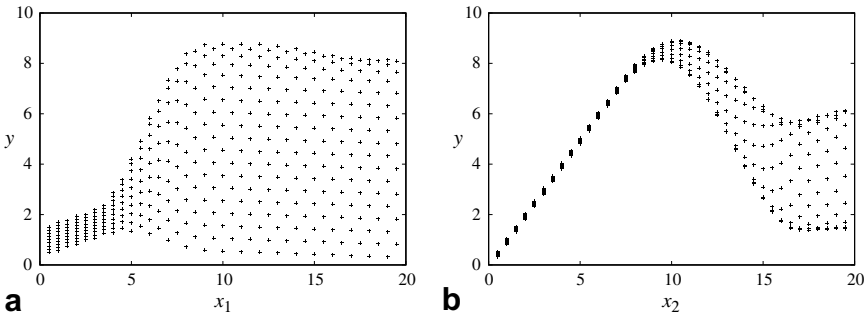


Fig. 4. SISO planes. (a) x_1–y plane with $x_2 \in [0, 10]$, (b) x_2–y plane with $x_1 \in [10, 20]$.

the shaded area in Fig. 3, we consider two typical angles represented as A and B in this figure. By assuming that input/output data were obtained at regular intervals over the input space, the x_1–y plane for A and x_2–y plane for B in which the data were plotted are shown in Fig. 4(a) and (b), respectively. In Fig. 4(a), we cannot determine specific information in the interval $[10, 20]$ because of the large spread of data points. On the other hand, Fig. 4(b) shows a high correlation between x_2 and y in the interval $[0, 10]$ because of the small spread of data points. This indicates that the input variable x_1 does not contribute to the output in the shaded area. Thus, angle B should be used to elicit a system feature effectively in the shaded area. In order to develop the features of the entire system, the IDS model is constructed based on four angles (A–D in Fig. 3).

An IDS unit processes data contained in an angle. It plots input/output data on its x_i–y plane ($1 \leqslant i \leqslant N$, N: number of inputs), and blurs the data points in the form of ink drop patterns, as illustrated in Fig. 5. This process is termed "data spread". As individual ink drop patterns overlap, the overlapping portions become increasingly darker; finally they form a pattern on the surface of the plane. The pattern can be in the form of continuous line(s) and/or spread(s). The continuous line exhibits a high correlation between the input and output; it is termed the "narrow path". The spread indicates that the output depends on other input variables. In the case of a large spread on an x_i–y plane, the input domains can be divided into smaller partitions to obtain a specific feature. In practice, the narrow path and spread are calculated in a manner that determines the mean and dispersion. The narrow paths and spreads are transferred from the IDS units to an upper processing layer

Fig. 5. Image of data spread in IDS.

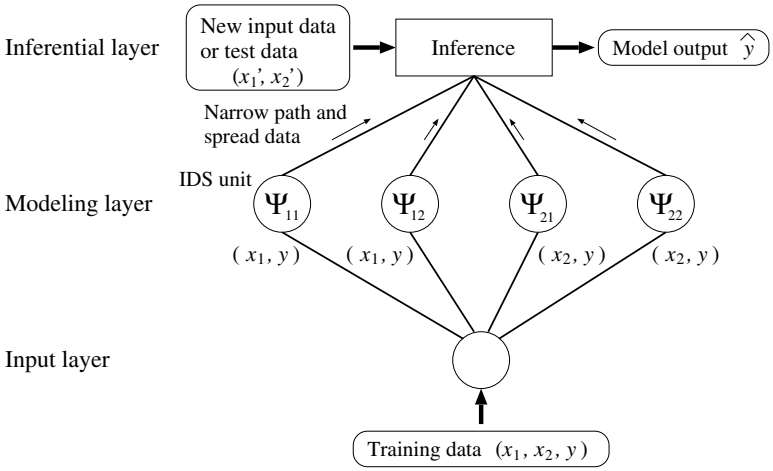


Fig. 6. Structure of an IDS model.

for an inferential process and are combined using a simple combination rule set in order to make predictions for new inputs.

In FNNs employing backpropagation learning (BPL) [8], information of input training data is transmitted over the network to the output layer, and backpropagation is performed in such a manner that the error between the network output and the value of the output training data is reduced by adjusting interconnection weights. This process in which the network conditions are varied to minimize the error is termed "learning". In the IDS method, we can construct a model without calculating the error between the model output and the value of the output training data. This is because training data are directly converted into useful information, namely, the narrow path and spread, for modeling. In addition, the IDS modeling does not require iteration of the same training set that is observed in the learning of neural networks. This direct model construction is a factor that enables real-time modeling. We define the process of generating a narrow path from training data (input/output data of a target system) as "learning" in the IDS method. Fig. 6 shows the structure of a two-input two-partition IDS model. The IDS model comprises three processing layers. The bottom input layer divides training data into SISO data and transfers them to the upper modeling layer. The top inferential layer computes a prediction using the learning data transferred from the IDS units. Unless a particularly high accuracy is required, the upper layer is not involved in the learning process of the IDS units. New input data for the prediction are directly fed to the inferential layer; thus, inferential and modeling processes are architecturally separated. This structural characteristic simplifies hardware realization and increases the scalability of the entire hardware system.

2.2. Computation in an IDS unit

An IDS unit executes the data spread on an x–y plane and computes the narrow path and spread from a pattern image formed on the plane. This section describes our implementation of the main part of the IDS unit.

Let $P_{xy} = \{p(x,y) | \ x \in X_i, y \in Y\}$ be the x–y plane, where $p(x,y)$ is the point (x,y) in the plane. $d(x,y)$ denotes the darkness at the point (x,y). When a data spread is applied at $p(x_s,y_s)$, the darkness of its neighboring area increases. We define this increased darkness as follows:

$$\begin{aligned} &\tau = R - \sqrt{u^2+v^2} + 1, \quad -R \leqslant u, v \leqslant R, \\ &\Delta d(x_s+u, y_s+v) = \begin{cases} [\tau], & \text{if } \tau > 0 \\ 0, & \text{otherwise,} \end{cases} \end{aligned} \tag{1}$$

which is the expression for an ink drop pattern with a radius R. $[\tau]$ denotes the greatest integer less than or equal to τ. The value of R must be fitted into the resolution of the x–y plane. In practice, R is determined not only from the resolution of the x–y plane but also through application requirements, such as the size of the training set, convergence speed, and model accuracy. In our standard setting, the size of the ink drop pattern is approximately 10% that of the x–y plane.

We used the bisector of area (BOA) method and a simplified method termed the "IDS α-cut spread" to obtain the narrow path and spread, respectively. The narrow path obtained by the BOA is represented by the following equation:

$$\psi(x) = \left\{ b \left| \sum_{y=1}^{b} \mathrm{d}(x,y) \approx \sum_{y=b}^{y_{\max}} \mathrm{d}(x,y), \quad b \in Y \right. \right\}, \tag{2}$$

where $y_{\max} = \max_{y \in Y} y$. When $\{y | \ \mathrm{d}(x,y) > \alpha, y \in Y\} \neq \phi$ for $^{\exists}x$, the IDS α-cut spread is defined by

$$\sigma_\alpha(x) = \max_{y \in Y} \{y | \ \mathrm{d}(x,y) > \alpha\} - \min_{y \in Y} \{y | \ \mathrm{d}(x,y) > \alpha\}. \tag{3}$$

Otherwise, $\sigma_\alpha(x)$ is substituted with the maximum value in the range. The IDS α-cut spread is based on the idea behind the α-cut of fuzzy sets.

In the inferential process, spread information is used to determine the certainty of the narrow path. In order to combine the narrow paths in the form of a weighted average, the spread is normalized by the following conversion function:

$$f_c(x) = \begin{cases} 1, & \text{if } \frac{\sigma_\alpha(x)}{y_{\max}} \leqslant P \\ \exp\left(-C\left(\frac{\sigma_\alpha(x)}{y_{\max}} - P\right)\right), & \text{if } P < \frac{\sigma_\alpha(x)}{y_{\max}}, \end{cases} \tag{4}$$

where $C > 0$ is the slope parameter of the exponential function and $P > 0$ is a criterion value used to determine whether certain patterns are narrow paths.

2.3. Inferential process

The IDS model computes the prediction using a simple combination rule set. We explain the inferential process using a two-input system comprising x_1 and x_2. When each input is divided into two partitions, the output y is determined using the following combination rule set:

$$\begin{aligned} R_{11}&: \text{ If } x_2 \text{ is } A_{21}, \text{ then } y_{11} \text{ is } \Psi_{11} \\ R_{12}&: \text{ If } x_2 \text{ is } A_{22}, \text{ then } y_{12} \text{ is } \Psi_{12} \\ R_{21}&: \text{ If } x_1 \text{ is } A_{11}, \text{ then } y_{21} \text{ is } \Psi_{21} \\ R_{22}&: \text{ If } x_1 \text{ is } A_{12}, \text{ then } y_{22} \text{ is } \Psi_{22} \end{aligned} \tag{5}$$

$$y \text{ is } \beta_{11}y_{11} \text{ or } \beta_{12}y_{12} \text{ or } \beta_{21}y_{21} \text{ or } \beta_{22}y_{22}, \tag{6}$$

where A_{ij} is the membership function that expresses the jth partition for the ith input variable. Ψ_{ik} represents the narrow path of the kth IDS unit for the ith input variable. The output of the IDS unit is obtained using (2), i.e., $y_{ik} = \psi_{ik}(x_i)$. β_{ik} denotes the weight of the output of the kth IDS unit for the ith input variable. In (6), the outputs of the IDS units are combined based on the weights determined using the degree of truth of the antecedent part in (5) and certainty of the narrow path given by (4).

2.4. Modeling characteristics

This section describes the characteristics of IDS modeling through the regression of the following nonlinear function:

$$y(x_1, x_2) = \sqrt{2\left(\frac{\sin x_1}{x_1}\right)^2 + 3\left(\frac{\sin x_2}{x_2}\right)^2}, \quad 1 \leqslant x_1, x_2 \leqslant 10. \tag{7}$$

Fig. 7 is a graphical representation of this function. For two-input functions, when input domains X_1 and X_2 are divided into m_1 and m_2 partitions, respectively, the number of IDS units used is $m_1 + m_2$. The number of partitions of the input domain directly affects the model accuracy. A few partitions may be sufficient to model a simple target function. However, for a complex target function, the number of partitions must be increased in order to represent its complexity. The increase in the number of partitions requires more training data. For example, let us assume that each IDS unit utilizes 25 data points to generate a narrow path. In such a case, the two-partition IDS model in which each input domain is divided into two partitions requires more than 50 training data points, while the six-partition IDS model requires more than 150 training data points.

In the regression modeling of (7), each input domain was divided into two, four, and six partitions at regular intervals. For each partition condition, we examined the effects of the

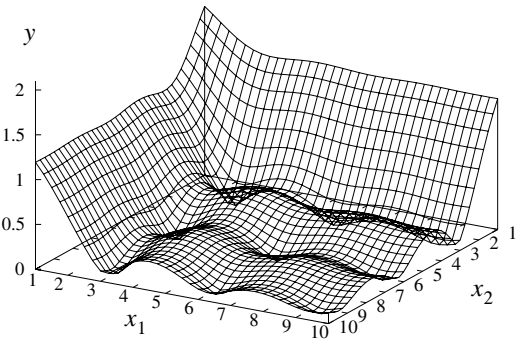


Fig. 7. Graph of (7).

Table 1
Three data spread sizes

Size of ink drop pattern	Radius R	Ratio of ink drop pattern size to 1024×1024 x–y plane size (%)
Small	25	5
Medium	51	10
Large	76	15

ink drop pattern size and training set size on the accuracy of IDS models. We tested three ink drop pattern sizes. Table 1 lists the values of R for the three sizes. Using (7), ten different training sets were generated randomly. Each training set contains 550 data points. In order to verify the model accuracy, the error between the target function and the constructed model was measured using the fraction of variance unexplained (FVU) [9] and the correlation coefficient. Let $\boldsymbol{x}_l\,(l = 1, 2, \ldots, L)$ be the lth input vector. The FVU is expressed as follows:

$$\mathrm{FVU} = \frac{\sum_{l=1}^{L} \left(\hat{y}(\boldsymbol{x}_l) - y(\boldsymbol{x}_l)\right)^2}{\sum_{l=1}^{L} \left(y(\boldsymbol{x}_l) - \bar{y}\right)^2}, \tag{8}$$

where $\hat{y}$ is the output of a constructed model and $\bar{y} = (1/L)\sum_{l=1}^{L} y(\boldsymbol{x}_l)$. The FVU is proportional to the mean squared error. As the model accuracy increases, the FVU approaches 0, and the correlation coefficient approaches 1. Each measure was calculated from 10,000 points at regular intervals over the input space. Let T_1 and T_2 be the sets of x_1 and x_2 coordinates, respectively. The input vectors of the test set are described by the Cartesian product of T_1 and T_2, which are obtained from

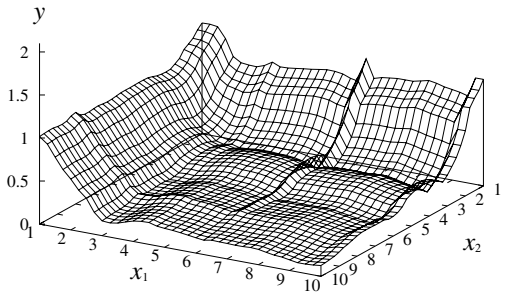
$$T_i = \left\{ t_i \middle| t_i = \frac{2p-1}{200} \times c_1 + c_2,\ p = 1, 2, \ldots, 100 \right\}, \quad i = 1, 2, \tag{9}$$

where c_1 and c_2 are constants determined by the input domain. From the domain of (7), $c_1 = 9$ and $c_2 = 1$. The values of FVU and correlation coefficient were recorded for four training set sizes: 100, 250, 400 and 550 data points.

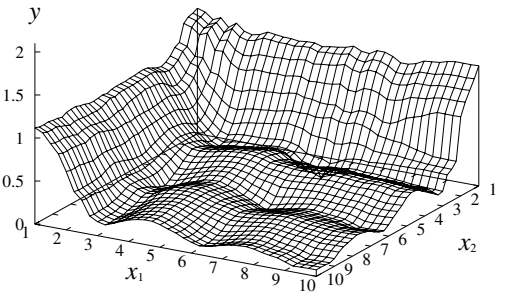
Table 2 lists the accuracy of each IDS model estimated by varying the number of partitions, the ink drop pattern size, and the training set size. The results revealed several important properties of IDS modeling. First, as the number of partitions decreases, the convergence speed in terms of the amount of training data used increases. In the measurement of the correlation coefficient, the two-partition IDS model, which exhibited high correlation, successfully represented the features of the target function. A small number of partitions are effective when the fast rough modeling is prioritized over high-accuracy modeling. Second, the model accuracy increases with the number of partitions when a sufficient amount of training data is used. The training set size is primarily responsible for the increase in the model accuracy. The accuracy of the six-partition IDS model was high, although it required a large amount of training data. Third, when the training set size is small, it is advantageous to use a large ink drop pattern instead of a small one. This is because the data spread corresponds to interpolation between data points; thus, a large ink drop pattern is effective in generating a narrow path with a small amount of training data. However, the use of a large ink drop pattern can result in the degradation of model accuracy. In regression modeling of 550 data points, the ink drop pattern of medium size

Table 2
Accuracy of each IDS model under different modeling conditions

Number of partitions	Ink drop pattern size	Training set size			
		100	250	400	550
FVU measure					
2	Small	0.378	0.124	0.088	0.077
	Medium	0.168	0.083	0.071	0.068
	Large	0.137	0.082	0.074	0.072
4	Small	0.756	0.159	0.063	0.042
	Medium	0.296	0.052	0.032	0.029
	Large	0.187	0.048	0.035	0.033
6	Small	1.049	0.251	0.093	0.046
	Medium	0.441	0.064	0.028	0.020
	Large	0.244	0.044	0.028	0.023
Correlation coefficient measure					
2	Small	0.803	0.975	0.989	0.990
	Medium	0.948	0.989	0.989	0.990
	Large	0.973	0.987	0.988	0.989
4	Small	0.670	0.923	0.972	0.983
	Medium	0.851	0.979	0.987	0.988
	Large	0.910	0.980	0.986	0.987
6	Small	0.622	0.883	0.955	0.979
	Medium	0.799	0.971	0.988	0.992
	Large	0.877	0.982	0.989	0.991



Two-partition IDS model
100 data points
Large ink drop pattern
FVU: 0.145
Correlation coefficient: 0.976



Six-partition IDS model
550 data points
Medium ink drop pattern
FVU: 0.020
Correlation coefficient: 0.994

Fig. 8. Output of sample models in regression modeling of (7).

produced slightly better results than the large ink drop pattern. Fig. 8 represents the output of two sample models. One is a two-partition IDS model that used a training set of 100 points and the large ink drop pattern. The other is a six-partition IDS model that used a training set of 550 points and the ink drop pattern of medium size.

3. Experiments

To investigate the modeling ability of the IDS method, we selected three benchmarks – the XOR problem, regression problem of Hwang's five-function set, and two-spiral problem – and compared the performance of the IDS method with that of the FNNs. In all experiments, each IDS unit used an x–y plane with a 1024×1024 resolution. The parameters of (4) were set as $C = 9$ and $P = 0.09$, and the parameter of the α-cut spread was set as $\alpha = 1$. The regression problem of Hwang's five-function set requires random numbers to generate the training data sets. We employed the drand48() function implemented as a pseudo-random number generator in the FreeBSD 5.4 operating system.

3.1. XOR problem

A modeling technique should be capable of not only dealing with complex nonlinear systems but substituting for simple logic operations. The XOR problem is the simplest learning problem that is not linearly separable. In FNNs employing BPL, even the construction of a simple XOR network requires repeated training with the four XOR patterns. In fact, we programmed a single-hidden-layer network with two hidden units based on BPL. Fig. 9 represents the output of two different XOR networks. The difference in the convergence of the two networks primarily depends on the initial conditions of interconnection weights. In order to perform binary logic operations by the IDS method, the input domain is divided by two. Data spreads on the x–y plane are applied at either $(0,0)$, $(0,1)$, $(1,0)$ or $(1,1)$. Fig. 10 represents the output of the IDS model constructed as an XOR function.

Neural networks can create different hypersurfaces between sparse training data points. As shown in Fig. 9, the response of a trained XOR is either approximately 0 or approximately 1 in the neighborhood of the point (0.5, 0.5). On the other hand, the IDS model can respond with a value of 0.5 or any arbitrary value in regions devoid of any training data. This is an important property for dealing with uncertainty.

3.2. Regression problem of Hwang's five-function set

Function approximation is a general tool used to evaluate the generalization performance of modeling techniques. Hwang et al. [9] first used the following five nonlinear functions g_i: $[0,1]^2 \rightarrow \Re$.

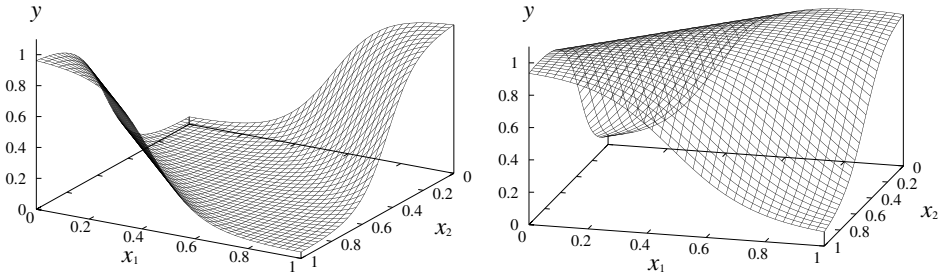


Fig. 9. Two different converged XOR networks.

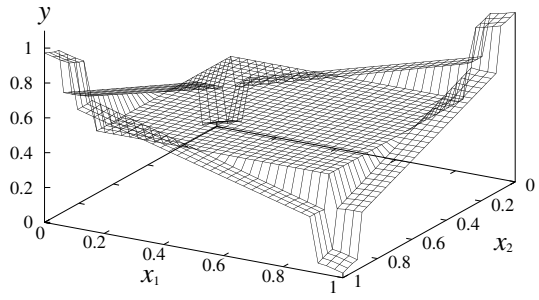


Fig. 10. Output of IDS model constructed as an XOR function.

- Simple interaction function:

$$g_1(x_1, x_2) = 10.391((x_1 - 0.4)(x_2 - 0.6) + 0.36).$$

- Radial function:

$$g_2(x_1, x_2) = 24.234(r^2(0.75 - r^2))$$
$$r^2 = (x_1 - 0.5)^2 + (x_2 - 0.5)^2$$

- Harmonic function:

$$g_3(x_1, x_2) = 42.659((2 + x_1)/20 + \mathrm{Re}(z^5))$$
$$z = x_1 + ix_2 - 0.5(1 + i)$$

or equivalently, with $\tilde{x}_1 = x_1 - 0.5$, $\tilde{x}_2 = x_2 - 0.5$.

$$g_3(x_1, x_2) = 42.659(0.1 + \tilde{x}_1(0.05 + \tilde{x}_1^4 - 10\tilde{x}_1^2\tilde{x}_2^2 + 5\tilde{x}_2^4)).$$

- Additive function:

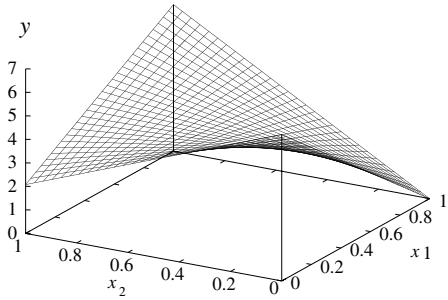
$$g_4(x_1, x_2) = 1.3356(1.5(1 - x_1) + e^{2x_1 - 1}\sin(3\pi(x_1 - 0.6)^2) + e^{3(x_2 - 0.5)}\sin(4\pi(x_2 - 0.9)^2)).$$

- Complicated interaction function:

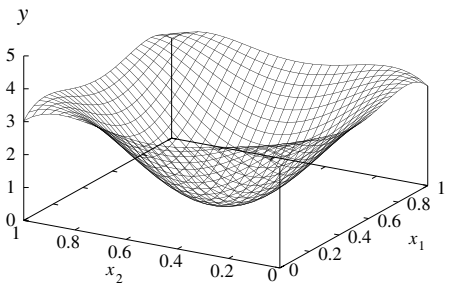
$$g_5(x_1, x_2) = 1.9(1.35 + e^{x_1}\sin(13(x_1 - 0.6)^2)e^{-x_2}\sin(7x_2)).$$

Fig. 11 graphically represents these functions. According to the standard test conditions of this benchmark, 225 random points were used as the training set and 10,000 uniformly distributed points were used as the test set. The test set can be obtained by setting $c_1 = 1$ and $c_2 = 0$ in (9). The model accuracy was measured using the FVU.

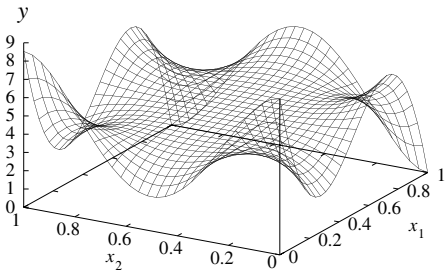
Many studies have compared the performance of FNNs using Hwang's five-function set [9–12]. Most of their benchmark results were based on the same test procedure described above. However, many authors used a single training set for learning each function. In such a case, the benchmark results can vary. In addition, they might have used a different data set. This indicates that it is not possible to determine the difference between the regression performances estimated in two studies by different authors. Hence, we used ran-



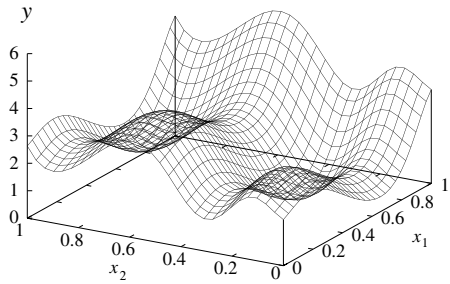
g_1: Simple interaction



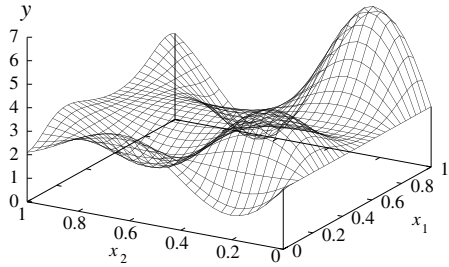
g_2: Radial



g_3: Harmonic



g_4: Additive



g_5: Complicated interaction

Fig. 11. Graphs of five functions. g_1, simple interaction; g_2, radial; g_3, harmonic; g_4, additive; and g_5, complicated interaction.

domly generated multiple training sets and verified our results by comparing them with those of other studies.

The FVU values cited from three different studies involving Hwang's five-function set are listed in Table 3. Hwang et al. [9] compared BPL and projection pursuit learning (PPL). In the benchmark, a batch Gauss–Newton method was adopted for the BPL and two types of PPLs were tested. One is supersmoother-based PPL, which is the standard PPL method, and the other is Hermite-polynomial-based PPL, which was proposed by Hwang et al. Kwok and Yeung [10] investigated objective functions for training new hidden units in constructive FNNs (CFNNs). Constructive algorithms begin with a small network and add hidden units while adjusting their weights until an optimal network is

Table 3
Accuracies of FNNs in the regression of the five-function set

Ref.	Model	Function				
		g_1	g_2	g_3	g_4	g_5
[9]	BPL based on Gauss-Newton method (5 hidden units)	0.001	0.065	0.506	0.080	0.142
	BPL based on Gauss-Newton method (10 hidden units)	0.001	0.002	0.183	0.003	0.021
	PPL supersmoother (3 hidden units)	0.000	0.010	0.355	0.021	0.135
	PPL supersmoother (5 hidden units)	0.000	0.007	0.248	0.000	0.028
	PPL Hermite (3 hidden units)	0.000	0.009	0.075	0.001	0.049
	PPL Hermite (5 hidden units)	0.000	0.000	0.000	0.001	0.015
[10]	CFNN with S_1	0.021	0.029	0.269	0.036	0.121
	CFNN with $\sqrt{S_1}$	0.011	0.028	0.247	0.037	0.111
	CFNN with S_2	0.095	0.426	0.547	0.636	0.610
	CFNN with $\sqrt{S_2}$	0.024	0.031	0.275	0.031	0.134
	CFNN with S_3	0.003	0.020	0.306	0.027	0.160
	CFNN with $\sqrt{S_3}$	0.003	0.018	0.288	0.030	0.167
	CFNN with S_{cascor}	0.025	0.027	0.265	0.031	0.121
	CFNN with S_{fujita}	0.004	0.047	0.444	0.070	0.246
	CFNN with S_{sqr}	0.007	0.038	0.573	0.185	0.294
[11]	Standard CFNN with sigmoidal activation functions (10 hidden units)	0.048	0.097	0.551	0.073	0.206
	Proposed CFNN with Hermite polynomial activation functions (10 hidden units)	0.031	0.027	0.197	0.076	0.095
	Standard CFNN with sigmoidal activation functions (20 hidden units)	0.043	0.048	0.303	0.050	0.111
	Proposed CFNN with Hermite polynomial activation functions (20 hidden units)	0.026	0.019	0.082	0.027	0.039

achieved. They tested nine objective functions (S_1, $\sqrt{S_1}$, S_2, $\sqrt{S_2}$, S_3, $\sqrt{S_3}$, S_{cascor}, S_{fujita}, and S_{sqr}) in the benchmark. These functions are described in [10]. They performed 100 independent trials in which different training sets were generated. In each trial, they recorded the lowest FVU among 15 networks ranging from 1 to 15 hidden units. The values listed in Table 3 are averages obtained from the 100 trials. Ma and Khorasani [11] studied a CFNN using Hermite polynomial activation functions. The activation functions of the hidden units in an FNN are usually the same; typically, they are sigmoidal or radial basis functions. In the proposed scheme, the activation functions of the hidden units are assigned successively from the lowest to the highest order orthonormal Hermite polynomials. Thus, the activation functions of each hidden unit are different. In the benchmark, the proposed CFNN was compared with a standard CFNN whose sigmoidal activation functions were identical.

In IDS modeling, we divided the input domain into many partitions in order to deal with complex functions; we tested six-, seven-, eight-, and nine-partition IDS models. However, 225 random data points are insufficient to generate narrow paths for these IDS models. Thus, we used the large ink drop pattern defined in Table 1, which is advantageous only when a small amount of training data is available. It should be noted that even by using the large ink drop pattern, we could not obtain complete narrow paths from 225 random data points. In this case, the optimal number of partitions is four or five.

Table 4
Accuracies of IDS Models in the regression of the five-function set

Number of partitions		Function				
		g_1	g_2	g_3	g_4	g_5
6	Mean	0.014	0.031	0.153	0.057	0.076
	Standard deviation	0.006	0.007	0.026	0.015	0.015
7	Mean	0.015	0.027	0.132	0.060	0.062
	Standard deviation	0.009	0.010	0.028	0.017	0.018
8	Mean	0.021	0.032	0.129	0.061	0.063
	Standard deviation	0.014	0.012	0.032	0.015	0.016
9	Mean	0.027	0.035	0.122	0.067	0.064
	Standard deviation	0.017	0.014	0.030	0.021	0.027

Table 4 shows the regression results of each IDS model. The values represent the means and standard deviations of the FVU values obtained from 20 trials using different training sets. The results of the FNN models in Table 3 indicate that it is particularly difficult to approximate functions g_3 and g_5, and the convergence of FNNs depends on the complexity of target functions. The accuracy of FNNs for simple functions is very high. In comparison with the FNN models, the accuracy of the IDS models is not significantly affected by the complexity of target functions. For the complex function g_3, the model accuracy in the IDS method was relatively high. In the regression of g_1, the model accuracy reduced with an increase in the number of partitions. This implies that the regression of g_1 does not require many partitions and the optimal number of partitions for 225 random data points is six or less. On the other hand, in the regression of g_3, the model accuracy increased with an increase in the number of partitions. Hence, it is concluded that the regression of g_3 requires many partitions due to its complexity. When an IDS model is constructed with an excess number of partitions and an insufficient amount of random training data, the model accuracy is affected by the uniformity of these data in the input domain. This is confirmed by the small and large standard deviations obtained for the six- and nine-partition IDS models, respectively.

In the IDS method, although regression modeling with a small amount of training data is disadvantageous, the results for 225 random data points were average in comparison with those of several FNNs. As described in [11], the learning parameters of FNNs must be predetermined in order that they can achieve the desired performance. In addition, random initial weights affect convergence. It is a significant advantage that the IDS model does not involve such intractability.

3.3. Two-spiral problem

The classification problem is a special case of the regression problem [10]. We evaluated the modeling ability of the IDS method in terms of classification. The two-spiral problem [13] is a well-known classification benchmark for supervised learning algorithms. In particular, FNNs have used the two-spiral problem to demonstrate the performance of new architectures and algorithms. The test involves the classification of two data sets that compose the two spirals. Typically, each spiral comprises 97 data points. Thus, 194 data points

are plotted on the plane. Points in the spiral are placed according to the following condition:

$$r = p(\theta + 2\pi n) + r_o, \tag{10}$$

where r and θ are the radius and angle, respectively. p and r_o are the parameters that determine the size of the spiral, and n is an integer that represents the number of revolutions. $p = 1/\pi$ and $r_o = 0.5$ are used as standard values.

Generally, single-hidden-layer networks based on standard BPL cannot perform stable perfect classifications in the two-spiral problem. Simple gradient descent methods involve numerous training iterations, which require a considerable amount of computational time. Only FNNs with refined architectures or algorithms can achieve 100% classifications with fast convergence. By using cascade-correlation learning proposed by Fahlman and Lebiere [14], it was possible to solve the two-spiral problem with fewer training iterations and a smaller network size. Hwang et al. [15] indicated the drawback of cascade-correlation learning that it is difficult to achieve a high accuracy in regression modeling tasks. Their projection pursuit learning network exhibited a good performance in both regression modeling and classification tasks. Jia and Chua [16] described the effect of input data encoding in which binary, weighted binary, gray, and temperature encoding schemes were tested. These encoding schemes can rapidly increase the number of input units. It is considered that the two-spiral problem can be solved by using many input units. Álvarez-Sánchez [17] proposed a knowledge-based neural network in which the radius and angle were used as inputs. This takes advantage of the geometric nature of input patterns; the use of polar coordinates is very effective in the classification of two spirals. Singh [18] used spiral data expanded into a three-dimensional coordinate space. In this case, the position of the spirals was determined by x, y, and z coordinates, and the z coordinate of the spirals was equal to the mean of the x and y coordinates. Singh used eight features including the x, y, and z coordinates as the input vector for the network. This expansion of spiral data into three-dimensional coordinate space does not necessarily complicate the classification of the two spirals because of the additional information on the z coordinate.

As described above, many researchers have attempted to determine an efficient solution to solve the two-spiral problem in FNNs by employing input encoding schemes or using preprocessing input data. It is evident that effective input data encoding improves the performance of FNNs. In other words, a supervised learning algorithm that can easily solve complex classification problems without adopting such techniques can serve as an excellent algorithm. Therefore, we applied the IDS method based on the standard use to the two-spiral benchmark, in the same manner as that of the previous benchmarks, without geometrically preprocessing the spiral data. In IDS modeling, each input domain was divided at regular intervals and the ink drop pattern of a medium size defined in Table 1 was used.

The IDS method cannot solve the two-spiral problem if the number of partitions of the input domain is insufficient. First, we examined the minimum number of partitions that could be used to achieve the perfect classification of the spiral data and found that the twelve-partition IDS model performed 100% classification. Fig. 12 shows the spiral data points separated into 12 partitions. When the model output at some data point was greater than 0.6, the data point was classified as a white spiral. When the model output was lower than 0.4, the data point was classified as a black spiral. The remaining range, 0.4 to 0.6, is defined as an invalid output. This rule has been frequently applied to the binary output of

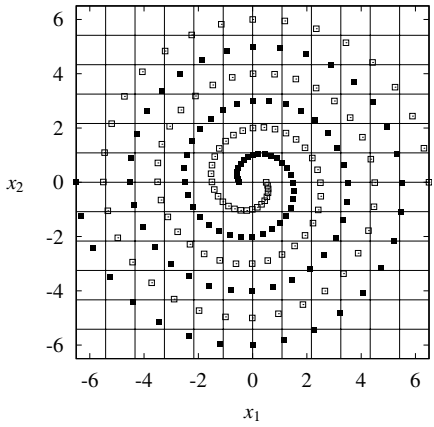


Fig. 12. Spiral data separated into 12 partitions for each input.

neural networks. Fig. 13 shows the output of the twelve-partition IDS model for 64×64 input points closely set at regular intervals over the data plane.

Next, we examined the generalization ability of the IDS model. We adopted the evaluation method of [18]. Four test sets were generated by introducing an offset in the training data: $(x_d + \delta, y_d + \delta)$, $(x_d + \delta, y_d - \delta)$, $(x_d - \delta, y_d + \delta)$, and $(x_d - \delta, y_d - \delta)$, where (x_d, y_d) is the spiral data point, and δ is the offset to be added; accordingly, the total number of test points is 776. In this experiment, the input domain is $[-6.5, 6.5]^2$. When a small offset is introduced in the original spiral data points, one or two test points in each test set are located outside the input domain. We excluded such points from the calculations of the classification rate.

Table 5 lists the average classification rates of the test sets for the IDS models. The number of partitions of each input domain was varied from 10 to 14. The results show that the generalization improved depending on the number of partitions. Thus, we infer that

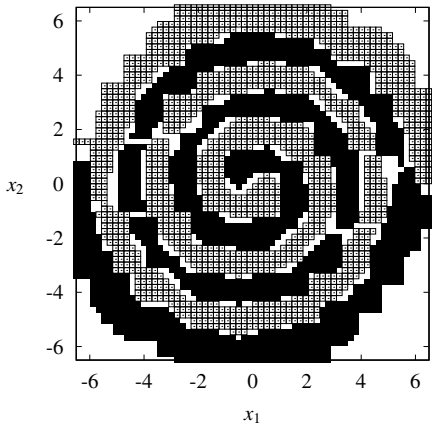


Fig. 13. Output of twelve-partition IDS model constructed using a two-spiral data set (100% classification achieved).

Table 5
Classification rate % of IDS models for the two-spiral benchmark

Offset δ	Test set size	Number of partitions				
		10	11	12	13	14
0	194	92.3	96.4	100	99.5	100
0.1	772	87.8	93.5	98.1	98.4	99.4
0.2	768	83.9	87.9	93.2	94.8	95.3
0.3	768	74.2	78.4	82.6	85.7	86.5

the generalization ability of the IDS method is sufficiently high in comparison with that of standard FNN models. For example, the average rates for the test sets with offsets 0.1, 0.2, and 0.3 in Singh's approach [18] were 96.5%, 91.0%, and 80.5%, respectively, although it was tested in a three-dimensional coordinate space. Since the IDS model does not use such randomly initialized parameters as the interconnection weights of FNNs, it can provide the same learning time and classification results for each trial.

In classification problems, the use of a large ink drop pattern will considerably reduce the classification performance of IDS models. This is because as the ink drop pattern size increases, wider ink drop patterns of different classes overlap at the same x coordinate on the x–y plane. By performing the same generalization test with the small ink drop pattern defined in Table 1, the average classification rates of the test data sets for the twelve-partition IDS model were found to be 99.0%, 94.7%, and 84.4% with offsets of 0.1, 0.2, and 0.3, respectively.

4. Conclusion

In this paper, we evaluated the modeling ability of the IDS method using three benchmarks. This method provides an ideal model for simple logic operations such as XOR. For solving the regression problem by using the training set of 225 random data points, the accuracy of the IDS model was satisfactory in comparison with the average generalization performance of FNNs. Further, the IDS method ensures an accurate model for a larger training set. This is evident from the modeling scheme of the IDS method, as described in Section 2.4. In the two-spiral problem, the classification performance of the IDS model was excellent. The twelve-partition IDS model could easily solve the two-spiral problem without using any geometric knowledge of the input data.

The benchmark results confirmed that the IDS method is an effective soft computing tool. Moreover, we reported in [6,7] that the IDS model possesses excellent real-time capabilities. For future study, we will apply the IDS method to practical real-time applications in order to validate its effectiveness.

References

[1] L.A. Zadeh, Soft computing and fuzzy logic, IEEE Software 11 (6) (1994) 48–56.
[2] A.B. Tickle, R. Andrews, M. Golea, J. Diederich, The truth will come to light: directions and challenges in extracting the knowledge embedded within trained artificial neural networks, IEEE Trans. Neural Networks 9 (6) (1998) 1057–1068.
[3] S.B. Shouraki, A novel fuzzy approach to modeling and control and its hardware implementation based on brain functionality and specifications, Ph.D. Thesis, The University of Electro-Communications, Chofu, Japan, 2000.

[4] S.B. Shouraki, N. Honda, Recursive fuzzy modeling based on fuzzy interpolation, Journal of Advanced Computational Intelligence and Intelligent Informatics 3 (2) (1999) 114–125.
[5] S.B. Shouraki, G. Yuasa, N. Honda, Fuzzy interpretation of human intelligence, International Journal of Uncertainty, Fuzziness and Knowledge-based Systems 7 (4) (1999) 407–414.
[6] M. Murakami, N. Honda, A study on the real-time modeling capabilities of the IDS method, Proceedings of the 14th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'05), Reno, Nevada, 2005, pp. 803–808.
[7] M. Murakami, N. Honda, Classification performance of the IDS method based on the two-spiral benchmark, Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics (SMC 2005), Hawaii, USA, 2005, pp. 3797–3803.
[8] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagationParallel Distributed Processing, vol. 1, MIT Press, Cambridge, MA, 1986, pp. 318–362.
[9] J.-N. Hwang, S.-R. Lay, M. Maechler, R.D. Martin, J. Schimert, Regression modeling in back-propagation and projection pursuit learning, IEEE Trans. Neural Networks 5 (3) (1994) 342–353.
[10] T.-Y. Kwok, D.-Y. Yeung, Objective functions for training new hidden units in constructive neural networks, IEEE Trans. Neural Networks 8 (5) (1997) 1131–1148.
[11] L. Ma, K. Khorasani, Constructive feedforward neural networks using Hermite polynomial activation functions, IEEE Trans. Neural Networks 16 (4) (2005) 821–833.
[12] J.-C. Duan, F.L. Chung, Cascade fuzzy neural network model based on syllogistic fuzzy reasoning, IEEE Trans. Fuzzy Systems 9 (2) (2001) 293–306.
[13] K.J. Lang, M.J. Witbrock, Learning to tell two spirals apart, Proceedings of the 1988 Connectionist Models Summer School, 1988, pp. 52–59.
[14] S.E. Fahlman, C. Lebiere, The cascade-correlation learning architecture, Tech. Report No. CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1991.
[15] J.-N. Hwang, S.S. You, S.R. Lay, I.C. Jou, The cascade-correlation learning: a projection pursuit learning perspective, IEEE Trans. Neural Networks 7 (2) (1996) 278–289.
[16] J. Jia, H.-C. Chua, Solving two-spiral problem through input data representation, Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, 1995, pp. 132–135.
[17] J.R. Álvarez-Sánchez, Injecting knowledge into the solution of the two-spiral problem, Neural Computing and Applications 8 (3) (1999) 265–272.
[18] S. Singh, Classification of helical structures, Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN 2000), Como, Italy, 2000, pp. 85–89.